PARKKIPAIKKASOVELLUS ANDROID-KÄYTTÖJÄRJESTELMÄLLE

HAMK
HÄMEEN AMMATTIKORKEAKOULU

Ammattikorkeakoulun opinnäytetyö

Tietojenkäsittelyn koulutusohjelma

Visamäki, kevät 2015

Pekka Pentonen

HAMK
HÄMEEN AMMATTIKORKEAKOULU

TIIVISTELMÄ

VISAMÄKI
Tietojenkäsittelyn koulutusohjelma
Systeemityö

Tekijä Pekka Pentonen **Vuosi** 2015

Työn nimi Parkkipaikkasovellus Android-käyttöjärjestelmälle

TIIVISTELMÄ

Opinnäytetyön toimeksianto tuli Ambientia Oy:ltä. Toimeksiantajalla oli tarve tehostaa Innoparkilta vuokraamiensa parkkipaikkojen käyttöastetta. Ratkaisuna ongelmaan toteutettiin parkkipaikkojen hallintaan tarkoitettu mobiilisovellus, jonka avulla yrityksen työntekijät voivat älylaitteillaan hallinnoida parkkipaikkoja.

Työn tavoitteena oli suunnitella ja toteuttaa natiivi Android-sovellus, joka kommunikoi REST-rajapinnan periaatteita noudattaen palvelinsovelluksen kanssa. Työn tuloksena on kehitetty sovellus ja opinnäytetyöraportti, jossa perehdyttiin työssä käytettyjen Android, ja REST-rajapinnan toteutustapoihin.

Opinnäytetyöraportin teoriaosuudessa esitellään työn toteutukseen vaadittavaa tietoa ja Android-ohjelmoinnissa vaadittuja seikkoja, jotka perustuivat Googlen laatimiin Android-kehittäjän ohjeistuksiin, suunnitteluperiaatteisiin ja suosituksiin. Aineistona käytettiin Professional Android 4 Application Developer -kirjaa.

Avainsanat Android, Java, ohjelmointi, järjestelmäarkkitehtuuri.

Sivut 26 s.

HAMK
UNIVERSITY OF APPLIED SCIENCES

Visamäki
Degree Programme in Business Information Technology

Author Pekka Pentonen **Year** 2015

Subject of Bachelor's thesis Parking Application for Android

ABSTRACT

This Bachelor´s Thesis was commissioned by Ambientia Oy which had a need to increase the utilization rate of parking spaces rented from Innopark. The aim of the thesis was to develop a mobile application that allows the employees of the company to manage the parking system using Android smart devices and design and implement a native Android application that uses the principles of the REST interface while communicating with the server application.

The theory part of the thesis discusses background information needed to carry out the study and information about Android programming. Android programming information is based on Google Android developer guidelines and Professional Android 4 Application Development book written by Reto Meier.

As a result of the thesis a mobile application was designed and a final report was drawn up including knowledge about how to implement an Android application and how to make it work with REST-interface.

Keywords Android, Java, programming, application architecture.

Pages 26 p.

SISÄLLYS

KÄSITELUETTELO

ANDROID
Android on avoimeen lähdekoodiin perustuva käyttöjärjestelmä.

Android SDK (Android Software Development)
Sovelluskehitysympäristö Android-käyttöjärjestelmälle, jossa kehityskielenä käytetään yleensä Javaa.

API (Application Programming Interface)
Ohjelmointirajapinta, jota hyödynnetään sovelluskehityksessä.

CORONA SDK (Corona Software Development Kit)
Usealle eri käyttöjärjestelmälle samanaikaisesti kehitettävän sovelluksen kääntämisen osaava ohjelmointiympäristö.

DDMS (Dalvik Debug Monitor Server)
Androidin ohjelmointiympäristössä käytetty työkalu, joka tuottaa tietoa kehitettävästä sovelluksesta.

FLUID UI
Verkkopalvelu, jota käytetään mobiilikäyttöjärjestelmien prototyypityksessä.

IDE (Integrated Development Environment)
Ohjelmointiympäristö, joka koostuu yhdestä tai useammasta sovelluksesta.

JDK (Java Development Kit)
Oracle Corporationin omistama ohjelmistokehityspaketti.

JSON (JavaScript Object Notation)
JSON on tiedon siirtämiseen ja tiedon kuvaukseen käytetty tiedostomuoto.

OHA (Open Handset Alliance)
Usean yrityksen liittouma, joka kehittää kännykkäalan avoimia standardeja ja vastaa muun muassa Android-käyttöjärjestelmän kehittämisestä.

PHP (Hypertext Preprocessor)
Ohjelmointikieli, jota käytetään dynaamisten verkkosivujen luonnissa.

REST (Representational State Transfer)
HTTP-protokollaan perustuva arkkitehtuurimalli, jonka periaatteena on sovelluksen tilan säilyttäminen asiakassovelluksella.

URI (Uniform Resource Identifier)
Merkkijono, joka kertoo sovelluksessa tai palvelimella sijaitsevan resurssin sijainnin verkkopolkuna tai hakemiston osoitteena.

XML (Extensible Markup Language)
Merkintäkieli, jota käytetään tiedon kuvaukseen tai formaattina tiedon tallentamiseen.

1 JOHDANTO

Tämän opinnäytetyön toimeksiantaja on Ambientia Oy. Yrityksessä työskentelee noin 120 työntekijää. Ambientia on asiantuntijayritys, joka auttaa asiakkaitaan konseptoimaan ja muotoilemaan sovelluksiaan sekä palvelujaan. (Ambientia Oy:n verkkosivut 2015)

Ambientian Hämeenlinnan toimistolla on noin 20 lämmityspistokkeellista parkkipaikkaa vuokrattuna Innoparkilta. Vuokratut parkkipaikat ovat nimettyjä osalle Ambientian työntekijöistä, eli parkkipaikan haltijalla on etuoikeus paikan käyttämiseen. Useat Ambientialla työskentelevät työntekijät matkustavat paljon työssään, jolloin osa parkkipaikoista on välillä vapaana työntekijöiden käytettäväksi. Tänä aikana parkkipaikkaa tarvitseva työntekijä voisi käyttää tyhjillään olevaa parkkipaikkaa. Ongelmana on, että parkkipaikkaa tarvitseva henkilö ei saa mistään tietoa, onko jokin parkkipaikka tyhjillään koko päivän.

Opinnäytetyön aiheena on tarjota ratkaisu parkkipakkojen varausjärjestelmään, eli suunnitella ja toteuttaa parkkipaikkojen käyttöasteen tehostamiseen tarkoitettu sovellus. Opinnäytetyön tuloksena syntyvä sovellus on tarkoitus ottaa alkuun Hämeenlinnassa sijaitsevan pääkonttorin käyttöön.

Opinnäytetyön työpaikkaohjaajana on Ambientialla sovelluskehittäjänä työskentelevä Tommi Kaisto. Opinnäytetyön ohjaajana toimii Hämeen Ammattikorkeakoulun tietojenkäsittelyn koulutusohjelman yliopettaja Tommi Lahti.

Sovelluksen suunnittelu ja toteutus tapahtuu osana kolmen tietojenkäsittelyn opiskelijan ryhmää. Kokonaisuutena tämä projekti käsittää kolme eri opinnäytetyötä, joista kukin opiskelija kirjoittaa oman raporttinsa, omasta näkökulmastaan. Tämä opinnäytetyö koostuu asiakassovelluksen suunnittelusta ja toteutuksesta Android-älylaitteille. Teemu Kivistö toteuttaa palvelimella ajettavan sovelluksen rajapinnan, johon mobiilisovellukset ottavat yhteyden. Marko Ojala tekee myös asiakassovelluksen, mutta käyttämällä Corona SDK:ta.

2 ANDROID SDK

Android Software Development Kit on ilmaiseksi saatavilla oleva kehitysympäristö, jolla voi kehittää Android-sovelluksia. Android SDK koostuu Javan sovelluskehittäjälle tarkoitetuista kirjastoista, Androidin kirjastoista ja kehitystyökaluista, jotka yhdessä muodostavat toimivan kehitysympäristön.

Sovelluskehitys Androidille on erittäin suosittua sen matalan oppimiskynnyksen ja kehitysympäristöjen ilmaisuuden ansiosta. Googlen julkaisemat oppaat ja dokumentaatiot ovat hyvin selkeitä ja helppolukuisia. Sovellusten julkaiseminen Google Play -kaupassa on kuitenkin maksullista ja vaatii kertaluontoisen rekisteröintimaksun, joka on 25 dollaria. (Google Inc. 2014b)

2.1 Historia

Android perustuu Linuxin ytimeen ja on kehitetty ensisijaisesti kosketusnäytöllisille älypuhelimille ja tableteille. Alun perin Android-käyttöjärjestelmää kehitettiin televisioille, pelikonsoleille ja kameroille vuonna 2003. Kehittäjä toimi alkuun Android Inc, jonka Google osti vuonna 2005. Nykyisin Androidin kehityksestä vastaa Open Handset Alliance ja julkaisijana toimii Google.

Google julkaisi ensimmäisen Android-ohjelmointirajapinnan vuonna 2008 (1.0, eli alpha versio). Ensimmäinen sovellusohjelmointia tukeva Android-ohjelmistorajapinta, eli API, julkaistiin huhtikuussa 2009 ja on nimeltään Cupcake, versionumerolla 1.5. Version merkittävimmät päivitykset olivat virtuaalisen näppäimistön ja videotallentamisen tuki. Versiosta 2.2, eli Froyosta lähtien Androidin virtuaalikone Dalvik alkoi tukemaan ajonaikaista kääntämistä, mikä lisäsi huomattavasti käyttöjärjestelmän suorituskykyä. Lisäksi Froyo sisältää tuen Open GL ES 2.0 rajapinnalle.

Ainoastaan taulutietokoneita tukeva Honeycomb API julkaistiin helmikuussa 2011 nimeltään versionumerolla 3.0. Honeycombissa uutuutena oli tuki moniydinsuorittimelle. Julkaisu on hieman erikoinen, koska Honeycombissa on tuki ainoastaan taulutietokoneille. Vuoden 2011 lopussa julkaistu 4.0, nimeltään Ice Cream Sandwich oli tärkeä julkaisu, koska se palautti yhteensopivuuden älypuhelimien ja tablettien kesken.

Jelly Bean, eli versiot 4.1, 4.2 ja 4.3 on julkaistu saman nimen alla. Versio 4.1 sisältää pääasiassa parannuksia käyttöjärjestelmän suorituskykyyn. Jelly Bean 4.2:n merkittävin uusi ominaisuus oli mahdollisuus luoda erilliset käyttäjätilit taulutietokone -laitteisiin. Versio 4.3 on julkaistu 2013, edelleen samalla Jelly Bean -nimellä. Kyseiseen versioon merkittävin uutuus on tuki Open GL ES 3.0 -rajapinnalle. Lisänä on muun muassa mahdollisuus vanhemmille rajoittaa lastensa käyttäjätilien pääsyä sovelluksiin sekä rajoittaa pääsyä verkkoon.

Versio 4.4, eli KitKat on julkaistu vuoden 2013 syyskuussa. Julkaisu sisältää suorituskyvyn optimoinnin laitteelle, jossa on vähemmän kuin 512 megatavua keskusmuistia.

Kuva 1. Android API versioiden nimen alkava kirjain määräytyvät aakkosten mukaisesti, alpha, beta, Cupcake ja niin edelleen. (Android version history 2014).

2.2 Java Development Kit

Android-sovelluskehitys tapahtuu Java-ohjelmointikielellä. Java on oliopohjainen ohjelmointikieli, jossa käytetään vahvaa tyypittämistä. Ohjelmointiympäristön rakentamiseen vaaditaan Java Developer Kit. Javan kirjastoja kehittää Oracle, ja kehityskirjastot on saatavilla ilmaiseksi Oraclen verkkosivustolta.

2.3 Kehitystyökalut

Suosituimpia kehitystyökaluja Androidille ovat Eclipse, IntelliJ IDEA, NetBeans ja Android Studio. Nämä kehitysympäristöt toimivat Linux-, Macintosh- ja Windows-käyttöjärjestelmissä. Kehitystyökalun lisäksi toimivan kehitysympäristön rakentamiseen vaaditaan Java SE Development Kit. Kaikki edellä mainitut työkalut ovat saatavilla ilmaiseksi.

2.4 Eclipse

Eclipse on avoimeen lähdekoodiin perustuva ohjelmointiympäristö. IBM on kehittänyt sovellusta vuodesta 1993. Sovellus on julkaistu avoimen lähdekoodin lisenssin alaisuudessa vuonna 2001. Nykyisin sovellusta kehittää Eclipse Foundation -niminen säätiö. Eclipseen saa lisättyä Android Developer Tools lisäosan, jonka avulla Android-kehitysympäristön saa rakennettua.

Ohjelmointiympäristössä sovelluksen voi kääntää suoraan Android-älypuhelimelle tai emuloiden vastaavaa laitetta. Eclipsen DDMS-työkalu tarjoaa mahdollisuuden seurata virtuaalisten laitteiden tiloja, tiedostojärjestelmää, säikeiden prosesseja ja muistin käyttöä.

Android 4.0:sta lähtien DDMS tarjoaa tuen myös verkkoliikenteen tarkkailuun. Työkalulla voi esittää yksittäisten sovellusten verkon käyttämää liikennettä pylväsdiagrammein ja samanaikaisesti laskea yksittäisten verkossa kulkevien pakettien koot ja määrät.

DDMS-näkymästä löytyvä LogCat on hyvä työkalu kehitystyössä tapahtuvien tapahtumien ja poikkeusten logittamiseen. Logeja voi määritellä tulostuvan eri tasoille, jolloin tekstit näkyvät eri värein. Logittamisen eri tasoja ovat verbose, debug, information, warning ja error. (Google Inc. 2015a)

2.5 Android Studio

Android Studiosta julkaistiin alpha versio maaliskuussa vuonna 2013. Beta versio julkaistiin kesäkuussa 2014. Android Studion kehittämisestä vastaa Google. Android Studio on korvannut suositun Eclipsen ja ADT-lisäosaan perustuvan ohjelmointiympäristön ja on siis Googlen suosittelema Android-kehitysympäristö. Android Studio perustuu IntelliJ IDEA:n lähdekoodiin.

Android Studion uutena ominaisuutena on Gradleen perustuva ohjelmakoodin kääntäminen. Gradle on joustava työkalu ohjelmakoodin kääntämisessä ja kääntämiseen vaadittavien riippuvaisuuksien hallinnassa. Kääntämisessä ei käytetä XML-kieltä, vaan lyhempään ja siistimpään syntaksiin perustuvaa Groovy-ohjelmointikieltä. Perinteisten jar-kirjastojen kääntämisessä Android Studio käyttää yhä Apache Ivya. (Google Inc. 2014a)

2.6 Intellij IDEA

Intellij IDEA on JetBrains yhtiön kehittämä ohjelmointiympäristö. Ensimmäinen versio IntelliJ IDEA:sta on julkaistu vuonna 2001 Apache 2 lisenssin alaisuudessa. Ohjelmointiympäristöstä on julkaistu kaksi eri versiota, maksullinen Ultimate Edition sekä ilmainen, ominaisuuksiltaan huomattavasti suppeampi Community Edition. Molemmat versiot tukevat kuitenkin Android-kehitysympäristöä. (JetBrains s.r.o website 2015.)

3 ANDROID OHJELMOINTI

Android-sovelluskehys tarjoaa sovelluskehittäjille rajapinnan, joka sisältää esimerkiksi Activity-managerin, Location Manager, eli GPS-anturin toiminnot, OpenGL ES, 3D-grafiikan piirtämiseen, SQLite-tietokannan ja niin edelleen. Näistä Activityt ja Fragmentit kuuluvat sovelluksen komponentteihin, jotka ovat Android-ohjelmoinnin perusteissa ensimmäinen vastaan tuleva asia.

3.1 Activityt

Activityt ovat niin sanottuja näkymä-luokkia, joissa suoritetaan varsinaiset käyttäjien ja sovelluksen väliset vuorovaikutukset. Androidilla näkymän luominen tapahtuu periyttämällä näkymäluokkaan Activity-niminen luokka. Jokainen Activity on esiteltävä sovelluksen Android Manifest-tiedostossa.

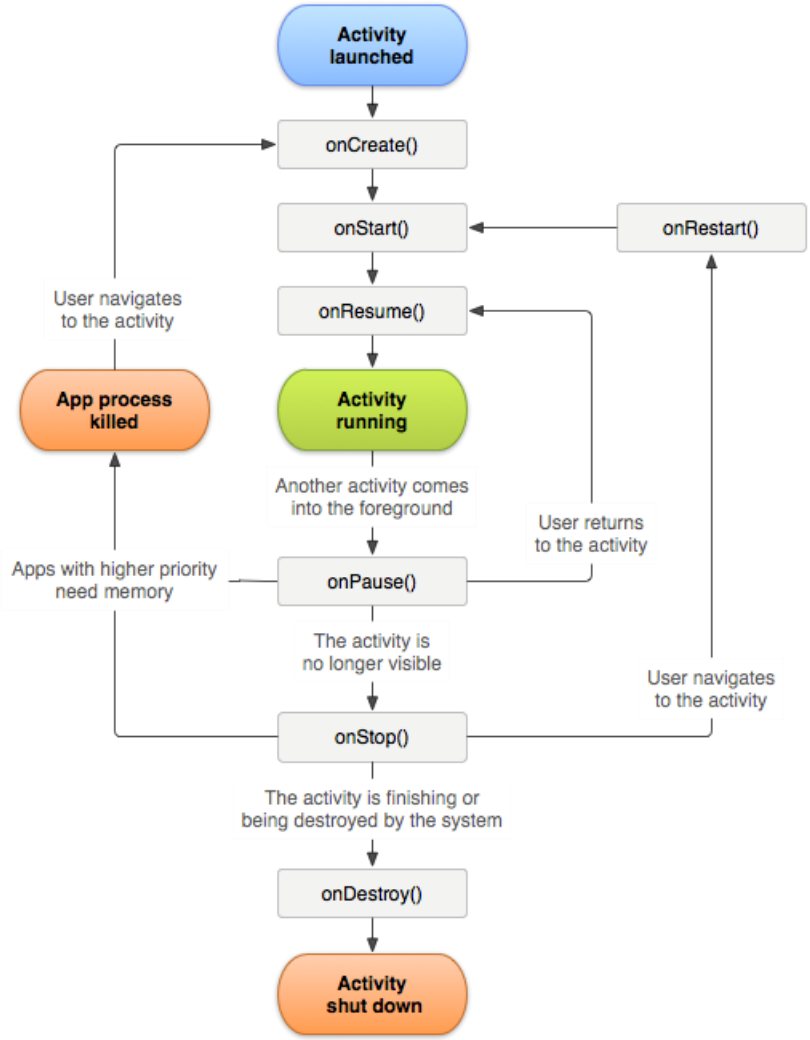
Androidilla ohjelmoidessa puhutaan käyttöliittymäsäikeestä, jossa ei ole tarkoitus ajaa raskaita toimenpiteitä. Raskaisiin toimenpiteisiin kuuluvat esimerkiksi tietokannan muokkaaminen tai tietokannasta tiedon lukeminen. Kaikki raskaammat tehtävät tulee ajaa tausta-ajona joko omissa palveluluokissa tai sisäluokissa.

3.2 Activityn elinkaari

Sovelluksessa näkymästä toiseen siirtyminen perustuu Intent-olioon. Intentin metodi nimeltä startActivity käynnistää uuden näkymän, jolloin edellisen näkymän tila siirtyy ensiksi taukotilaan. Edellinen näkymä pysähtyy kun uusi näkymä piirtyy laitteen näytölle, jolloin onStop metodi on suoritettu.

Uuteen näkymään siirryttyä Android kutsuu automaattisesti onDestroy metodia, joka tuhoaa edellisen näkymän lopullisesti, joka tarkoittaa sitä, että sen käytössä oleva muisti vapautuu. Joskus tulee tarve kutsua onRestart metodia, joka käynnistää Activitystä uuden instanssin ja hävittää kaikkien muistissa olevien muuttujien arvot.

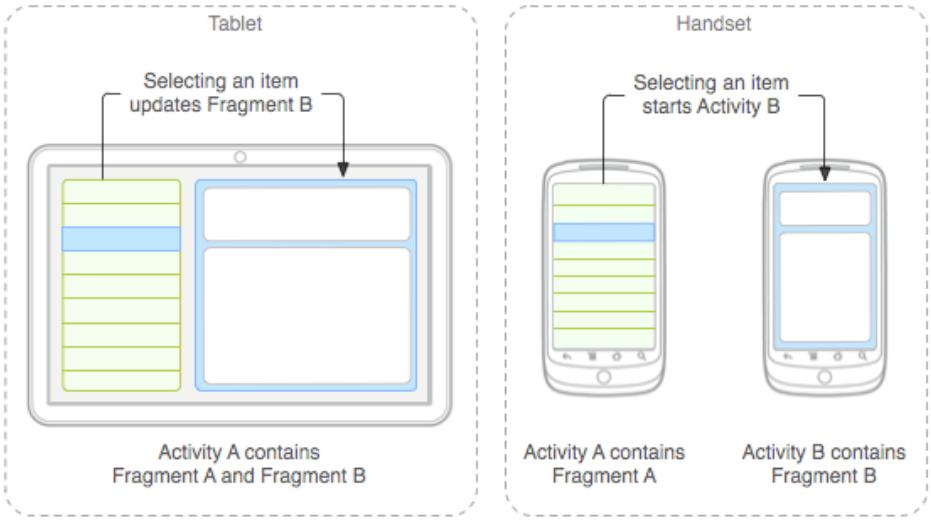
Activitylla on elinkaari, jonka tilan hallintaan on 7 eri metodia. Metodeissa toteutetaan eri tapahtumia



Kuva 2. Activityillä on eri tilat. Tärkeimmät tilat ovat kuvassa värillisenä. (Google Inc. 2014c.)

3.3 Fragmentit

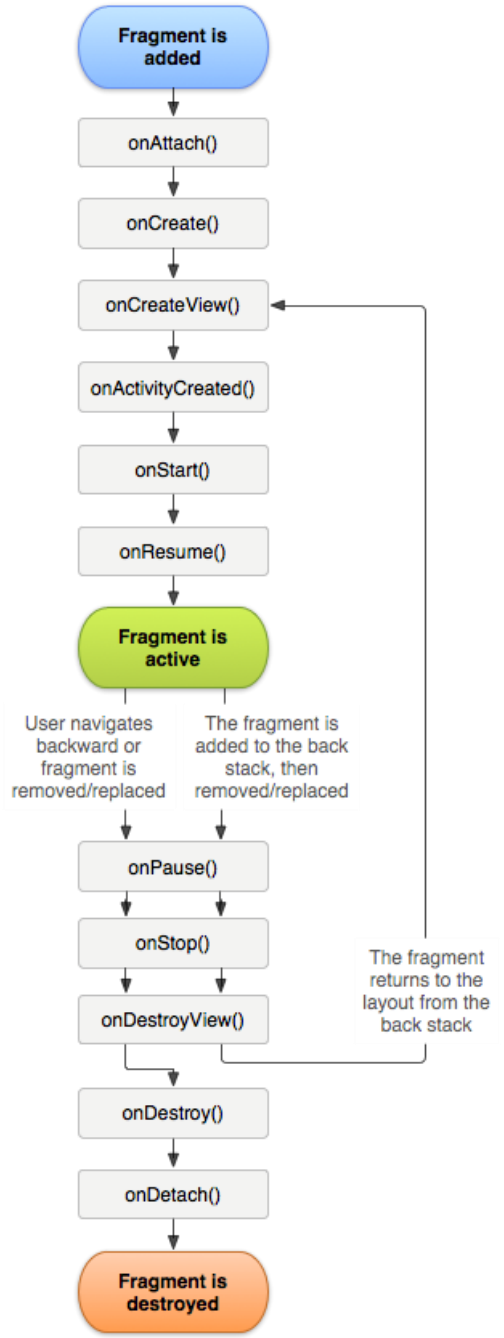
Fragmentit ovat Androidin API 3.0 -versiosta lähtien tarjonneet tuen isommalla näytöllä varustetuille taulutietokoneille. Fragmentit ovat Activityn sisällä toimivia moduuleja ja jokaisella moduulilla on oma elinkaarensa. Taulutietokoneissa on suuri näyttökoko, joka voidaan tunnistaa laskien laitteen näytön leveys ja asettaa ruudulle kaksi fragmenttia rinnakkain joissa molemmissa on oma toiminnollisuutensa.



Kuva 3. Tabletille mahtuu kaksi fragmenttia rinnakkain. Puhelimessa on vain yksi fragmentti kerrallaan. (Google Inc. 2014d.)

3.4 Fragmentin elinkaari

Jokaisella Fragmentilla on vastaava elinkaari kuin Activityillä.



Kuva 4. Fragmentin elinkaari. (Google Inc. 2014d.)

3.5 Käyttöliittymä

Android-sovelluskehyksessä käyttöliittymien elementit ja niiden ominaisuudet määritellään XML-tiedostojen avulla. Elementtejä ovat esimerkiksi painikkeet, listat, valikot ja tekstikentät. Elementin sijainti, tyyli tai toiminnot määritetään antamalla elementeille eri attribuutteja.

Kaikki XML-tiedostot sijaitsevat projektin res-hakemiston alaisuudessa. Käyttöliittymän elementit otetaan käyttöön alustaen ne Activity-luokissa, jonka jälkeen elementeille voi määritellä haluttuja toimintoja.

3.6 Kolmannen osapuolen kirjastot

Erilliset kirjastot ovat apuna monimutkaisia sovelluksia toteuttaessa, kun Android Developer Toolsin rajat tulevat vastaan. Osa kolmannen osapuolen kirjastoista on dokumentoitu puutteellisesti, jolloin niiden käyttöönottaminen on haastavaa. Erillisiä kirjastoja käyttäen päästään yleensä nopeammin haluttuun lopputulokseen, kunhan eri vaihtoehtoihin ensin tutustuu huolellisesti.

Android Volley-kirjasto tarjoaa kehittyneitä ominaisuuksia tiedonsiirtoon Androidissa. Esimerkiksi http-pyyntöjä voi asettaa jonoon ja priorisoida suoritettavaksi ensisijaisesti verraten vähemmän tärkeisiin kyselyihin. Pyyntöjä voi ryhmittää nimeten ne samalla nimellä ja peruuttaa joko yksittäisen tai ryhmään kuuluvien pyyntöjen suorittaminen, esimerkiksi aktiviteettiä käynnistäessä uudelleen.

Pyynnöt voi kirjoittaa suoraan käyttöliittymä-säikeeltä, koska Volley osaa suorittaa ne asynkronoidusti oletusarvoisesti. JsonObjectRequest lähettää ja palauttaa dataa JSON-oliona. JsonObjectRequest metodi tukee POST, GET, DELETE ja PUT-pyyntöjä. JsonArrayRequest metodi lähettää tiedon JSON-taulukkona ja tukee tällä hetkellä vain GET-pyyntöä.
(Google Inc. 2015b.)

TimesSquare for Android -kalenteri on Squaren kehittämä kirjasto, joka tarjoaa edistyneemmän kalenterikomponentin kehittäjän käyttöön. Sama kalenterikomponentti on saatavilla Androidille ja iOS:lle. TimesSquare kalenterissa on perustoimintoina toteutettu yksittäisen päivämäärän valitsin, päivämäärien moni valinta ja laajempi päivämäärien aluevalinta, jolloin valinnan voi tehdä alkaen ensin valitusta päivästä päättyen seuraavaksi valittuun päivämäärään. (Square Inc. 2014)

4 REST

REST tarkoittaa tilatonta arkkitehtuurimallia. Tilaton arkkitehtuuri tarkoittaa sitä, että yhteyden tilaa ei tallenneta REST-rajapinnalle, vaan sen toteutus jää asiakassovellusten vastuulle. Mallin periaatteita käytetään toteuttaessa rajapintaa, jonka kanssa asiakassovellukset kommunikoivat.

REST-mallissa on määritelty jokaiselle resurssille oma yksilöllinen URI, eli osoite jolla rajapintaan otetaan yhteys. Toiminnoille määritellään osoitteen lisäksi parametreja, jotka muuttavat resurssien tilaa tietomallissa. Parametrien avulla resursseja voi luoda, poistaa ja päivittää. Lukemisellekin voi määritellä parametrien avulla että mitä tietoa luetaan. Resurssien lukeminen on ainoa metodi, joka ei muuta resurssien tilaa.

REST:in arkkitehtuuriin lukeutuvat seuraavat metodit: GET, POST, PUT, PATCH, OPTIONS ja DELETE. Jokaiseen edellä mainittuun metodiin vaaditaan otsikko, eli HEADER. GET-metodia käytettäessä vaaditaan vähintään Accept-otsikon merkintä. Muut metodit vaativat tämän lisäksi myös Content-Type otsikon. OPTIONS-metodia voidaan käyttää pyytäessä listaa rajapinnalta, sen käytössä olevista metodeista.

4.1 METODIT

CRUD-operaatiot, eli luo (create), lue (read), päivitä (update) ja poista (delete). Nämä operaatiot ovat tietokantojen käsittelyssä käytettyjä metodeja. Operaatioita kutsutaan palvelimella riippuen http-kyselyjen tyypistä ja niille asetetuista parametreista.

REST toteuttaa nämä operaatiot käyttäen HTTP-protokollan kyselykutsuja nimeltään: POST, GET, PUT ja DELETE. Rajapinta voidaan määritellä hyväksymään vain osan edellä mainituista metodeista käsiteltäväkseen. Mikäli pyyntö on esimerkiksi GET-tyyppinen, niin toiminto lukee yksittäisen tiedon tietokannasta ja palauttaa sen kyselyn vastauksena. Samaan osoitteeseen saapuva POST-pyyntö voidaan ohjata eri metodille, jolla on täysin eri toiminto. Tässä tapauksessa POST-pyyntö ohjattaisiin metodille, joka suorittaisi POST-pyynnön parametrien tallentamisen tietokantaan.

4.2 URI

REST-arkkitehtuuri perustuu URI-osoitteisiin. Osoite yksilöi palvelimen resurssin sijainnin ja nimen, joihin asiakasohjelmat ottavat yhteyden. Osoitteet hyväksyvät ennalta määritettyjä http-pyyntöjä. Jokaisella http-pyynnöllä voi olla eri merkityksensä, vaikka URI on sama.

4.3 TIEDONSIIRTO

Lyhyesti sanottuna XML ja JSON ovat tiedonsiirrossa ja tiedon kuvaamisessa käytettyjä formaatteja. Näistä JSON on yksinkertaisempi kirjoittaa ja se on helpommin luettavissa. JSON on myös kevyempi tiedonsiirtoformaatti verrattuna XML:n syntaksiin.

```
<ilmoitukset>
    <ilmoitus>
        <id>1</id> <lot_nro>131</lot_nro>
    </ilmoitus>
    <ilmoitus>
        <id>2</id> <lot_nro>127</lot_nro>
    </ilmoitus>
    <ilmoitus>
        <id>3</id> <lot_nro>134</lot_nro>
    </ilmoitus>
</ilmoitukset>
```

Kuvio 1. Esimerkki XML:n käytöstä

```
{"ilmoitukset":[
    {"id":1, "lot_nro":131},
    {"id":2, "lot_nro":127},
    {"id":3, "lot_nro":134}
]}
```

Kuvio 2. Esimerkki käyttäen JSON:ia.

JSON-tietoa voidaan kuvata kahdella eri rakenteella, jotka ovat JSON-object ja JSON-array. JSON-oliot koostuvat avain-arvo pareista, joissa avain määritellään tekstinä ja avaimen arvon tyyppinä voi esiintyä joko teksti, luku, tosi-epätosi, tyhjä, olio tai taulukko. JSON-array on vastaavasti avain-arvo-pareista koostuva taulukko. JSON-tiedon MIME-media tyyppi on määriteltävä http-pyynnön otsikossa nimellä application/json, jotta tiedostoa lukeva sovellus, esimerkiksi verkkoselain osaisi tulkita mitä tietoa se ottaa vastaan ja miten sitä tulisi käsitellä. (w3schools 2015)

5 VERSIONHALLINTA

Versionhallinta on järjestelmä, jolla voi tallentaa tiedostojen eri työvaiheet. Tallennettuja tiedostoversioita voi käydä myöhemmin muokkaamassa. Yleisesti on käytössä kolme eri versionhallintajärjestelmää, jotka ovat paikallinen-, keskitetty-, sekä hajautettu versionhallintajärjestelmä.

Nykyisin ohjelmointityössä käytetään yleisesti hajautettua versionhallintajärjestelmää, koska se ei tarvitse yhteistä palvelinta, kuten keskitetyssä järjestelmässä. Hajautettu versionhallintajärjestelmä perustuu tietojen peilaamiseen useille eri koneille. Tiedostojen varmuuskopiot ovat palautettavissa jokaiselta järjestelmään peilatulta koneelta.

5.1 Git

Git on avoimeen lähdekoodiin perustuva versionhallintaohjelmisto. Gitin on suunnitellut ja kehittänyt Linus Torvalds vuonna 2005 kehittäessään tunnettua avoimen lähdekoodin käyttöjärjestelmää Linuxia. Gitin toiminta-ajatuksena on se, että muutosten tekeminen on nopeaa. Git-versionhallintajärjestelmää kehitetään yhteisöpohjaisesti. Muun muassa Android-käyttöjärjestelmä on kehitetty käyttäen Git-järjestelmää. (GitHub 2015.)

5.2 Bitbucket

Bitbucket on ilmainen versionhallintapalvelu viidelle käyttäjälle. Palvelu on avattu vuonna 2008 ja sen omistaa Australialainen teknologiayritys nimeltään Atlassian. Palvelu on alusta lähtien tukenut Mercurial-versionhallintaohjelmistoa ja vuodesta 2011 lähtien se on myös tarjonnut tuen Git-ohjelmistolle. (Atlassian Bitbucket website 2015.)

5.3 SourceTree

SourceTree on Atlassian kehittämä työpöytäsovellus, joka helpottaa versionhallintaa muun muassa Bitbucket-palvelussa. Ohjelma tarjoaa kaikki toiminnot ilman komentojen kirjoittamista perinteiseen konsoliin. Sovellus on ilmaiseksi saatavilla uusimmille Windows ja Mac OS X-käyttöjärjestelmille. (Atlassian SourceTree website 2015.)

6 SOVELLUS

Parkkipaikkasovelluksen ideana on tarjota työkalu yrityksen käyttöön, jolla voi tehostaa yrityksen omistamien tai vuokraamiensa parkkipaikkojen käyttöastetta. Parkkipaikkojen ilmoittaminen vapaaksi ja parkkipaikkojen varaamiset tapahtuu mobiililaitteilla. Parkkipaikkojen nimeämiset käyttäjille tapahtuu erillisessä ylläpitonäkymässä, johon on pääsy vain sovelluksen pääkäyttäjällä. Sovelluksen pääkäyttäjä voi hallinnoida verkkokäyttöliittymällä käyttäjiä ja parkkipaikkoja.

Sovelluksen käyttöönoton jälkeen on mahdollista kirjautua sovellukseen. Kirjautuminen sovellukseen toteutetaan käyttäen yrityksen toimialueen tunnistuspalvelinta. Kirjautuessa sovellukseen älylaitteella tapahtuu käyttäjän tunnistaminen ja kirjautuneen käyttäjän parkkipaikkatietojen, nimen ja sähköpostiosoitteen pyyntö palvelimelta. Sovelluksen käyttäjällä on kaksi eri roolia. Käyttäjä voi olla parkkipaikan omistaja tai käyttäjä jolle ei ole nimettynä parkkipaikkaa. Kirjautuessa sovellukseen käyttäjän tiedot kysytään palvelimelta, jolta palautuu vastaus, että onko käyttäjälle nimitetty parkkipaikkaa vai ei.

Vain käyttäjä joka omistaa parkkipaikan, voi ilmoittaa parkkipaikkansa vapaaksi viikonpäiville jolloin hän ei tarvitse sitä. Parkkipaikan vapaaksi ilmoittaminen toimii valitsemalla päivämäärä sovelluksen kalenterinäkymästä ja hyväksymällä varmistus ilmoitustapahtumasta. Sovelluksen kalenterinäkymästä omistaja voi myös muokata tai poistaa tekemänsä ilmoituksen.

Mikäli työntekijä on varannut vapaaksi ilmoitetun parkkipaikan ja paikan haltija haluaa muokata tai poistaa ilmoituksensa, niin paikan varanneelle käyttäjälle lähtee automaattisesti tieto sähköpostitse varauksen peruuntumisesta.

Mikäli käyttäjälle ei ole nimetty omaa parkkipaikkaa, niin sovelluksen näkymä tarjoaa ainoastaan muiden työntekijöiden vapaaksi ilmoitettujen parkkipaikkojen selaamisen, varaamisen ja tekemiensä varausten perumisen. Tehdyt varaukset listautuvat automaattisesti varauspäivän perusteella, nykyhetkestä tulevaisuuteen, eli nousevassa järjestyksessä. Käyttäjälle tulee listanäkymä vapaista parkkipaikoista, josta valitsemalla voi tehdä varauksen. Sovellus kysyy käyttäjältä varmistuksen ennen varaustapahtuman hyväksyntää. Parkkipaikkoja voi varata vain yhden paikan päivää kohden.

Näkymässä joka listaa omat varaukset onnistuu myös niiden peruminen. Varaamansa parkkipaikan peruutus näkyy varaajalle sovelluksessa omien varausten listanäkymässä niin, että varaus ei ole aktiivinen ja sitä ei pysty muokkaamaan. Parkkipaikan käyttöönsä tarvitseva työntekijä voi tarkistaa järjestelmästä onko vapaaksi ilmoitettua paikkaa saatavilla sinä päivänä jolloin hän tarvitsee sellaisen käyttöönsä.

6.1 Vaatimusmäärittely

Yhteisen suunnitelman pohjalta asiakassovellukseen toteutettavaksi määritettyjä toimintoja on yhteensä yksitoista kappaletta. Näitä ovat vapaiden parkkipaikkojen listaus, parkkipaikan varaaminen, parkkipaikan varauksen peruminen, oman ilmoituksen päivämäärän muokkaus, omien varauksien listaus, omien ilmoituksien listaus, parkkipaikan ilmoittaminen vapaaksi, ilmoituksen poistaminen, käyttäjän tunnistaminen, kirjautuneen käyttäjän tietojen noutaminen ja käyttäjän kirjautuminen ulos.

Mikäli käyttäjä on varannut parkkipaikan, jonka paikan haltija on ilmoittanut, on paikan haltijalla oikeus peruuttaa ilmoitus. Tässä tapauksessa paikan varanneelle käyttäjälle on tultava ilmoitus sähköpostitse, että hänen varauksensa on peruuntunut. Sähköpostin lähetys tapahtuu rajapinnalta.

Rajapinnalle on toteutettava myös ylläpitonäkymä, mutta se ei vaikuta mobiilisovelluksiin millään tavalla, koska ainoastaan sovelluksen verkkokäyttöliittymään tulee ylläpitäjän toiminnot. Vain ylläpitäjällä on oikeus lisätä järjestelmään parkkipaikkoja, hallita parkkipaikkojen omistajuuksia, sekä poistaa parkkipaikkoja.

Vaatimuksena on, että sovellukseen voi kirjautua yrityksen toimialueen tunnistuspalvelinta hyödyntäen, jolloin työntekijöille ei tarvitse luoda erikseen uusia tunnuksia parkkipaikkasovellusta varten. Mobiilisovelluksen toteutetaan kaksi eri näkymää, joista aina toinen piirtyy automaattisesti parkkipaikan omistajuuden perusteella sovellukseen kirjatuessa. Jos käyttäjällä on parkkipaikka omistuksessa, niin hän voi ilmoittaa vapaaksi haluamilleen päiville kalenterinäkymästä.

Kun parkkipaikaton käyttäjä kirjautuu sisään, voi hän selata vapaita parkkipaikkoja listanäkymästä ja tehdä varauksen valitsemalla paikan listasta. Jos samalle päivälle on useampi vapaa parkkipaikka, niin näytetään listassa myös vapaiden parkkipaikkojen määrä.

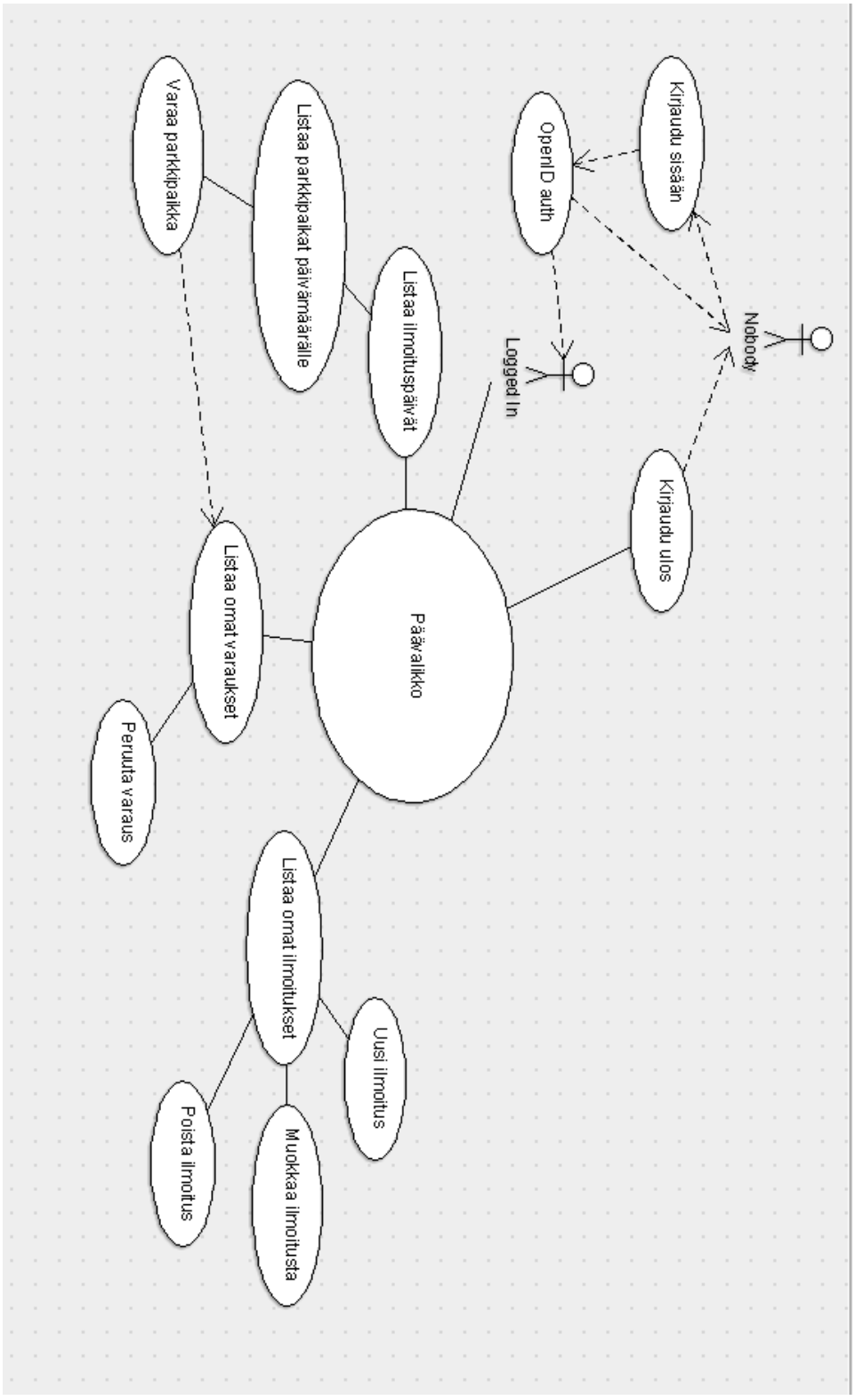
Parkkipaikkavarauksen peruuttaminen onnistuu listasta, johon listautuvat kaikki omat varaukset. Parkkipaikaton käyttäjä saa tehdä vain yhden varauksen päivää kohden. Käyttöliittymälle ei asetettu tiukkoja vaatimuksia, joten työssä keskitytään enemmän sovelluksen toimintojen toteuttamiseen kuin sovelluksen ulkoasuun. Käytettävyys otetaan kuitenkin myös huomioon jo suunnitteluvaiheesta lähtien.

6.2 Käyttöliittymä

Alustava käyttöliittymäproto tehdään Fluid UI:lla, joka toimii mallina sovelluksen käyttöliittymän toteutukselle. Käytettävyyttä miettien voisi hyödyntää eri näyttökokojen tarjoamia ominaisuuksia. Jos sovellusta käytetään esimerkiksi tabletilla älypuhelimen sijaan, niin esimerkiksi vapaita parkkipaikkoja selatessa voidaan tulostaa molemmat tiedot samalla kertaa isolle näytölle.

6.3 Käyttötapausten kuvaukset

Käyttötapauskaavio on hyvä apu ohjelmoidessa suunniteltuja toimintoja sovellukseen. Kaaviosta kuvataan kaikki sovelluksessa vaaditut toiminnot ja niiden hierarkia.



Kuva 5. Asiakassovelluksen käyttötapaukset.

6.4 Testaus

Testauksessa pyritään ottaa huomioon mahdolliset ongelmakohdat tai loogiset ohjelmointivirheet. Testausta ei ole mahdollista käydä läpi pelkästään kehittäjien toimesta, koska testitapauksien kirjoitus vaiheessa itse sokaistuu testaamaan joko liian yksityiskohtaisia asioita tai ei niinkään käytettävyyden kannalta tärkeitä asioita vaan enemmänkin suorituskykyä parantavia koodikutsujen sijaintia ja järjestystä.

6.5 Julkaisu

Sovellus vaatii erillisen palvelimen, jossa parkkipaikkojen varaustiedot sijaitsevat. Palvelimen täytyy tukea salattua http-protokollaa, jotta sovellusta olisi turvallista käyttää. Palvelimen julkaisun jälkeen asiakassovellukseen määritetään julkaisupalvelimen osoite, josta se vastaa asiakassovellukset pyyntöihin. Jatkossa työntekijät pystyvät lataamaan sovelluksen omalle älylaitteelleen yrityksen sisäverkossa sijaitsevalta verkkosivustolta, jonka osoite ja ohjeistus sovelluksen käyttöönottoon ilmoitetaan yrityksen työntekijöille esimerkiksi sähköpostitse.

6.6 Jatkokehitys

Myöhemmin sovellusta voisi kehittää käytettäväksi esimerkiksi yrityksen muissa toimipisteissä. Sovelluksen jatkokehityksenä voisi pitää muun muassa sitä, että käyttäjän varaamaan parkkipaikkaan olisi ajo-opastus Google Maps-kirjastoja hyödyntäen.

Sovelluksen käyttöliittymään toteutettaisiin erilliset toiminnot taulutietokoneille. Tämä toteutus vaatisi sovelluksen osittaisen toteuttamisen käyttäen Androidin Fragmentteja. Uusi toiminnollisuuksia olisi esimerkiksi parkkipaikkojen varauksien selaaminen siten, että vasemmalla puolella näkymää olisi lista vapaista parkkipaikoista ja oikealla puolella näkymää tulostuisi samanaikaisesti tiedot valitusta varauksesta.

Parkkipaikan varannutta käyttäjää helpottaisi pohjakuva parkkialueesta, jossa paikkojen numerot ovat esillä. Pohjakuvasta voisi varata vapaana olevan parkkipaikan klikkaamalla parkkipaikan ruutua, jossa olisi myös parkkipaikan numero esillä.

7 TULOKSET

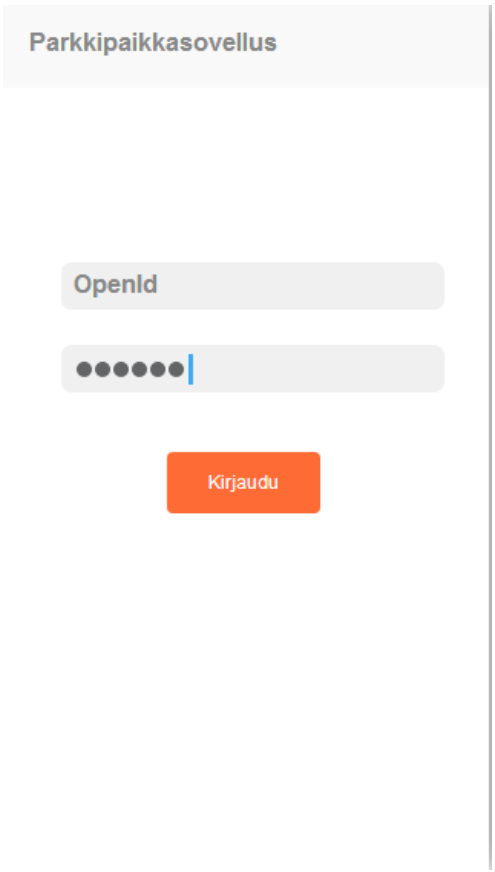
Projektin tuloksena on sovellus, jossa on toteutettuna kaikki vaatimusmäärittelyissä määritetyt toiminnollisuudet. Sovellus määritettiin tukemaan Androidin käyttöjärjestelmää versiosta 3.0 lähtien.

Sovelluksen toteutusta lähdettiin suunnittelemaan kolmen opiskelijan tiiminä, Tommi Kaiston ohjauksessa. Sovellukseen vaadittavat toiminnot ja tekniikat käytiin yhdessä läpi. Laadimme sovelluksen alustavan tietokantasuunnitelman ja kävimme keskustelua siitä, miten sovellus täyttäisi sille asetetut vaatimukset.

.

Alustava käyttöliittymäsuunnitelma on tehty käyttäen Fluid UI -palvelua.

Kuva 6. Parkkipaikkasovelluksen alustavan käyttöliittymäproton näkymä, jossa listataan vapaat parkkipaikat. Vapaiden parkkipaikkojen määrä on mukana listauksessa.



Kuva 7. Sovelluksen kirjautumisikkuna

Kuva 8. Sovelluksen päävalikko.

Sovelluksen toteutus oli loogista aloittaa käyttöliittymän elementtien sijoittelusta käyttöliittymäprototyypin mallista ja edetä sen jälkeen käyttöliittymän toiminnollisuuksien toteuttamiseen.

Sovellusta toteuttaessa päädyin käyttämään ohjelmointiympäristönä Android Developer Tools -pakettia, johon sisältyi Eclipse ja Android Developer Tools. Eclipse oli projektin alkaessa vakain ympäristö ja aiemmista projekteista tuttu. Kokeilin myös uudempaa Android Studiota, mutta mielestäni sovellus oli vielä hieman puutteellinen käytettävyydeltään ja toiminnoiltaan.

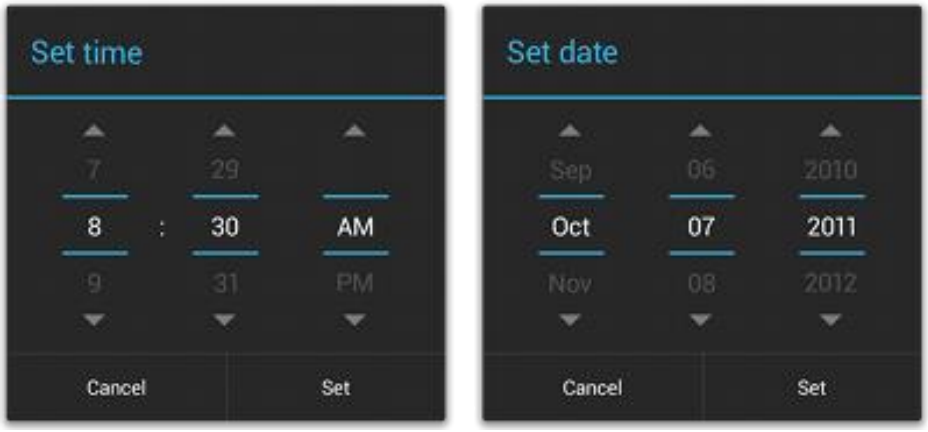
Suunnitellessa sovelluksen arkkitehtuuria, sovimme yhdessä projektin jäsenten kanssa, että kaikki tieto siirtyy JSON-taulukkoina rajapinnan ja asiakassovelluksen välillä. Päätös pitäytyä ainoastaan JSON-taulukoissa oli hyvin kriittinen, jotta jatkossa välttyisimme suuremmilta yhteensopivuus ongelmilta. Projektin alussa sovelluksen kehittämisen apuna oli myös testirajapinta, joka on toteutettu käyttäen PHP-ohjelmointikieltä. Testirajapinta oli hyvä apuväline kehittäessä perustoimintoja, kuten esimerkiksi vapaiden parkkipaikkojen listaamista. Testirajapinnan avulla oli mahdollista testata sovellukseen suunniteltua tietokantaa ja sen rakenteen toimivuutta käytännössä.

Kokeiltaessa Android Volley-kirjastoa taulukkojen parserointiin ja http-pyyntöjen lähettämiseen, tuli kirjaston vajavaisuudet pian vastaan. Android Volley ei tukenut kuin http-get metodia lähettäessään kyselyn rajapinnalle, joka palauttaa JSON-taulukon vastauksena. Koska olimme projektiryhmän kanssa aiemmin sopineet vain JSON-taulukoiden käytöstä, niin Volley-kirjaston käyttö piti unohtaa.

Päädyin toteuttamaan JSON-taulukkojen parseroinnin Androidin omilla kirjastoilla käyttäen AsyncTaskia ja apuluokkaa. Tässä toteutuksessa apuluokasta tehdään olio AsyncTask-sisäluokassa, jolloin sen metodit saadaan käyttöön. Metodeja kutsutaan AsyncTaskin sisältä, jolloin niiden suoritus tapahtuu a-synkronoidusti. Apuluokan makeHttpRequest-metodi vaatii toimiakseen rajapinnan URI-osoitteen, http-metodin tyypin ja optiona se ottaa vastaan listan parametreista ja tekstimuodossa kyselyn otsikon. Pyynnön vastauksena saatu JSON-taulukko iteroidaan toistolauseella, jolloin päästään haluttuihin tietoihin käsiksi. (Meier. 2012, 345-347.)

Sovellukseen kirjautumisen toteutus oli yksi projektin haastavimmista vaiheista. Sovellukseen kirjautumisessa oli alkuun tarkoitus hyödyntää OpenID-tunnistustapaa, joka käyttää julkisia tunnistuspalvelimia tunnistaessaan käyttäjän kirjautumista sovellukseen. Tästä suunnitelmasta kuitenkin luovuttiin, sillä kyseiselle tekniikalle ei löytynyt hyvää dokumentaatiota ja ohjeistusta Corona SDK:lle.

Lopulta käyttäjän tunnistaminen sovellukseen toteutettiin tekniikalla, joka perustuu rajapinnalla generoitavaan merkkijonoon, eli tokeniin. Jokaisen käyttäjän yksilöivä token luodaan käyttäjän kirjautuessa sisään onnistuneesti. Token palautuu rajapinnalta asiakassovellukseen onnistuneen kirjautumispyynnön vastauksena ja samalla se tallennetaan asiakassovelluksen muistiin. Jatkossa jokaisen http-pyynnön otsakkeessa lähetetään asiakassovelluksen muistista luettu token, jota verrataan palvelimella olevaan tokeniin.



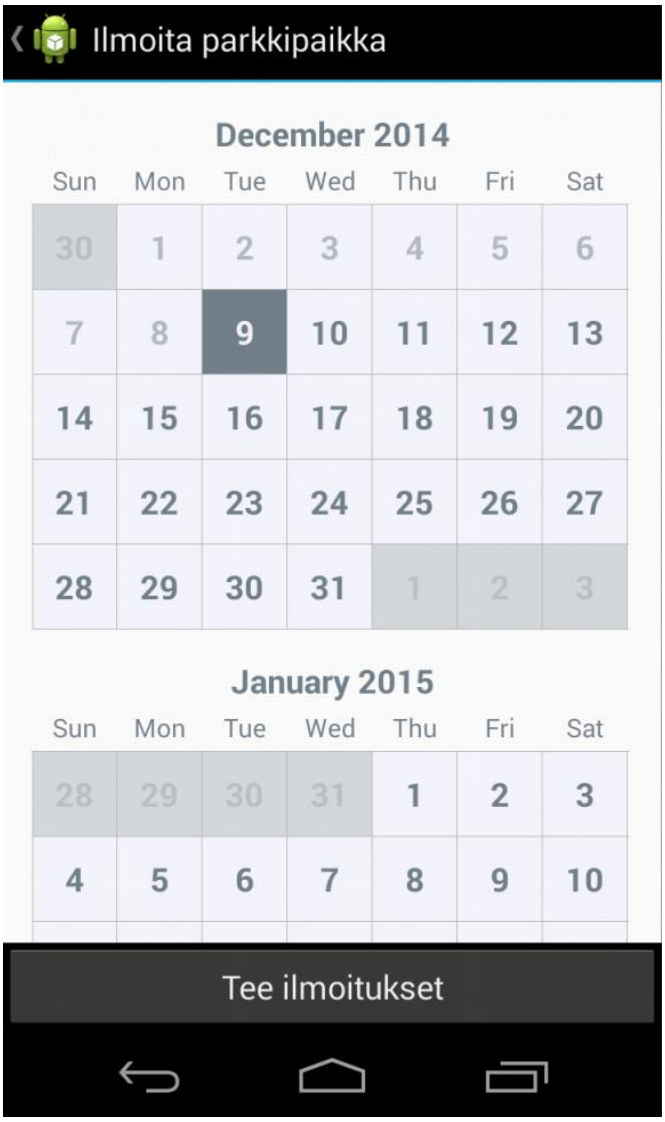
Kuva 9. Androidin päivämäärävalitsin. (Google Inc. 2014e.)

Parkkipaikan ilmoittaminen vapaaksi toiminnon ensimmäinen toteutusversio perustui päivämäärävalitsimeen, joka löytyy Androidin kirjastoista. (Kuva 9). Tämä toteutus oli toimiva, mutta käytettävyydeltään hieman kömpelö, koska parkkipaikkoja pystyi ilmoittamaan vapaaksi vain yhden kerrallaan. Tässä vaiheessa sovelluksen käytettävyyttä tuli arvioida uudelleen ja miettiä sovelluksen loppukäyttäjiä enemmän.

Tämän johdosta päätimme tutkia mahdollisuutta hyödyntää kalenterinäkymää, josta käyttäjä voisi ilmoittaa parkkipaikkansa vapaaksi useammalle päivälle kerrallaan. Kyseisen toiminnon toteutukseen ei Androidin oman kalenterikomponentin ominaisuudet riittänyt ja siksi monipuolisemman kalenterin toteuttamiseen on käytetty Squaren Androidille kehittämää kalenterikomponenttia nimeltä Android TimesSquare.

7.1 Käyttöliittymä

Kalenterinäkymässä parkkipaikan haltija voi ilmoittaa parkkipaikan vapaaksi valitsemalla kalenterista yksittäisen päivämäärän tai useita päivämääriä kerrallaan, jonka jälkeen ilmoitustapahtuma hyväksytään alalaidassa olevasta painikkeesta.



Kuva 10. Kuvassa on parkkipaikkojen ilmoittamisen kalenterinäkymä.

Kuva 11. Valmiin sovelluksen päävalikko

Vapaat parkkipaikat näkymässä listautuvat päivät, jolloin vapaita parkkipaikkoja on saatavilla. Valitsemalla päivänmäärän listasta, aukeaa seuraava näkymä, josta voi valita parkkipaikan ja tehdä varauksen. Listanäkymä on toteutettu Androidin ListView-elementillä, johon ilmoitusten tiedot luetaan taulukosta, joka sisältää kaksi listaa. Listat sisältävät päivämäärän ja vapaiden parkkipaikkojen määrän vastaavaa päivää kohden.

Kuva 12. Kuvassa on toteutetun sovelluksen vapaiden parkkipaikkojen listanäkymä.

7.2 REST-rajapinta

Sovellukseen toteutettiin lopulta yksitoista erilaista REST-rajapintaa kutsuvaa metodia. GET pyyntöä on käytetty vapaiden parkkipaikkojen listan noutamisessa rajapinnalta. Sovellus käyttää REST-arkkitehtuurin periaatteita kommunikoidessaan palvelimen kanssa, lukuun ottamatta http delete-pyyntöä, joka on lopulta korvattu käyttäen http post-pyyntöä. Post pyyntö on kuitenkin tässä tapauksessa väärä toteutustapa, joka on jatkokehityksen myötä muutettava delete-pyynnöksi. Tämä selkeyttää sovelluksen arkkitehtuuria ja tekee siitä helpommin dokumentoitavan. Listatessa vapaita parkkipaikkoja on käytetty http-get pyyntöä.

Parkkipaikan varaaminen ja varauksen peruminen toimii http-put pyynnöllä. Uusi ilmoitus vapaasta parkkipaikasta lähetetään post-pyyntönä, jonka parametreina on tieto siitä, kuka omistaa parkkipaikan ja mille päivälle parkkipaikka ilmoitetaan vapaaksi.

7.3 Kirjautuminen

Sovellukseen kirjautuminen tapahtuu syöttämällä ylläpidon toimesta luotu käyttäjätunnus ja salasana. Tietojen syöttämisen jälkeen käyttäjä painaa kirjaudu sisään painiketta, joka lähettää tunnuksen ja salasanan tarkistuspyynnön rajapinnalle. Mikäli käyttäjätunnus ja salasana täsmäävät rajapinnalla luodun käyttäjätunnuksen kanssa, niin se generoi asiakkaalle oman, yksilöivän tokenin, joka palautuu rajapinnalta vastausviestinä.

Jos käyttäjätunnukset eivät täsmää palvelinsovelluksen tietojen kanssa, niin kirjautuminen epäonnistuu ja epäonnistumisesta palautuu viesti käyttäjälle.

8 YHTEENVETO

Opinnäytetyön tavoitteena oli suunnitella ja toteuttaa toimeksiantajan parkkipaikkojen hallintaa tehostava mobiilisovellus Androidille. Työn teknisessä toteutuksessa tutkittiin sitä, miten Android-sovelluksen saa kommunikoimaan REST-rajapinnan kanssa. Tämän mobiilisovelluksen toteutus oli osa yhteistä projektia, jossa kolme tietojenkäsittelyn opiskelijaa tekivät kukin omaa osuuttaan.

Opinnäytetyölle asetettuihin tavoitteisin päästiin ja opinnäytetyön tuloksena syntyi toimiva sovellus. Työtä aloittaessa minulla ei ollut lainkaan kokemusta REST-arkkitehtuurimallista tai sen toiminnasta. Minulta löytyi aiempaa kokemusta Android-sovelluksen toteutuksesta, jossa oli käytetty ainoastaan http-lukupyyntöjä.

Opinnäytetyö opetti uutta asiaa liittymän toteuttamisesta Android Developer Tools -kirjastoilla, sekä osittain myös ulkoisia kirjastoja apuna käyttäen. Huomioitavaa on myös se, että ulkoisia kirjastoja käyttäessä ohjelmakoodin määrä vähenee huomattavasti ja sovelluksen suorituskyky paranee. Haasteena ulkoisten kirjastojen käyttöönotossa on niiden niukka dokumentaatio, eikä esimerkkikoodeja löydy montaakaan.

Opinnäytetyö antoi hyvää kokemusta ryhmätyöskentelystä ja opetti versionhallinnan käyttämisen tärkeyden ohjelmointityössä. Versionhallinta helpottaa ja nopeuttaa työskentelyä huomattavasti, vaikka siinäkin on omat haasteensa alkuun.

LÄHTEET

Ambientia Oy:n verkkosivut 2015. Viitattu 20.5.2015. http://www.ambientia.fi/fi/suomen-paras-liferay-portaalien-ja-atlassian-ratkaisujen-asiantuntija

Google Inc. 2014a. Viitattu 6.12.2014. https://developer.android.com/sdk/index.html

Google Inc. 2014b. Viitattu 9.12.2014. https://developer.android.com/distribute/googleplay/start.html

Google Inc. 2014c. Activities. Viitattu 16.12.2014. http://developer.android.com/guide/components/activities.html

Google Inc. 2014d. Fragments. Viitattu 21.9.2014. http://developer.android.com/guide/components/fragments.html

Google Inc. 2014e. Pickers. Viitattu 12.11.2014. http://developer.android.com/guide/topics/ui/controls/pickers.html

Android version history 2014. Wikipedia. Viitattu 2.10.2014. http://techviha.com/android-os-version-history/

JetBrains s.r.o website 2015. IntellJ IDEA. Viitattu 24.3.2015. https://www.jetbrains.com/idea/features/editions_comparison_matrix.html

Google Inc. 2015a. ADT plugin release notes. Viitattu 16.3.2015. http://developer.android.com/tools/sdk/eclipse-adt.html

Google Inc. 2015b. Transmitting Network Data Using Volley. Viitattu 5.2.2015. http://developer.android.com/training/volley/index.html

Square Inc. 2014. Android TimesSquare widget. Viitattu 10.10.2014. http://square.github.io/

Atlassian SourceTree website 2015. Free Git & Mercurial client for Windows or Mac. Viitattu 2.2.2015. https://www.sourcetreeapp.com/

w3schools 2015. JSON tutorial. Viitattu 23.5.2015. http://www.w3schools.com/json/

Atlassian Bitbucket website 2015. Git and Mercurial code management for teams. Viitattu 23.5.2015. https://bitbucket.org/

GitHub 2015. Alkusanat versionhallinnasta. Viitattu 15.5.2015. http://git-scm.com/book/fi/v1/Alkusanat-Versionhallinnasta

Meier, R. 2012. Professional Android 4 Development. John Wiley & Sons, Inc.